Peter Mitura
miturpet@fit.cvut.cz
presented:

Simple and efficient fully-functional succinct trees

Joshimar Cordova, Gonzalo Navarro
https://arxiv.org/pdf/1601.06939.pdf

Basics

Definition 1. An *ordinal tree* is a rooted tree of arbitrary degree in which the children are ordered.

Lemma 1. It is possible to represent any ordinal tree of n nodes using less than $2n$ bits of space.

The goal of this paper is to show an easy to implement succinct representation of ordinal trees that supports wide range of operations (as shown in Table 1). $\mathcal{O}(\log\log n)$ operation time is achieved, using $2n + \mathcal{O}(n \,/\, \log n)$ bits of space.

Primitive operations

Given a bitvector $B[1, 2n]$ representing the tree:

- $rank_t(i)$ $t \in \{0, 1, 10\}$ – number of occurencies of bit(s) t in $B[1, i]$.
- $select_t(k)$, $t \in \{0, 1, 10\}$ – position of k-th occurence of t in B.
- *(helper)* $excess(i) = rank_1(i) - rank_0(i) = 2rank_1(i) - i$
- *(helper)* $close(i) = \min\{j \mid j > i,\, excess(j) = excess(i) - i\}$
- *(helper)* $open(i) = \max\{j \mid j < i,\, excess(j-1) = excess(i)\}$
- *(helper)* $enclose(i) = \max\{j \mid j < i,\, excess(j-1) = excess(i) - 2\}$
- $fwdsearch(i, d) = \min\{j \mid j > i,\, excess(j) = excess(i) + d\}$
- $bwdsearch(i, d) = \max\{j \mid j < i,\, excess(j) = excess(i) + d\}$
- $rmq(i, j)$: position of the leftmost minimum in $excess(i), excess(i+1), \ldots, excess(j)$.
- $rMq(i, j)$: position of the leftmost maximum in $excess(i), excess(i+1), \ldots, excess(j)$.
- $mincount(i, j)$: number of occurences of the minimum in $excess(i), excess(i+1), \ldots, excess(j)$.
- $minselect(i, j, q)$: position of the qth minimum in $excess(i), excess(i+1), \ldots, excess(j)$.

Please note that helper operations only serve for abstraction, and are implemented using other primitives.

Range min-max trees

Given a block size b, range min-max tree of bitvector $B[1, 2n]$ is a complete binary tree where the kth leaf covers $B[(k-1)b+1, kb]$. Each rmM-tree node v stores the following fields:

- $v.e$ – total excess of the area covered by v
- $v.m$ – minimum excess in this area
- $v.M$ – maximum excess in this area
- $v.n$ – number of times minimum excess occurs in this area

Operation	**Description**	**Implementation**
$root$	the tree root	1
$preorder(i)$	preorder rank of node i	$rank_1(i)$
$postorder(i)$	postorder rank of node i	$rank_0(close(i))$
$preorderselect(k)$	the node with preorder k	$select_1(i)$
$postorderselect(k)$	the node with postorder k	$open(select_0(k))$
$isleaf(i)$	whether the node is a leaf	iff $B[i+1]=0$
$isancestor(i,j)$	whether i is an ancestor of j	iff $i \leq j < close(i)$
$depth(i)$	depth of node i	$excess(i)$
$parent(i)$	parent of node i	$enclose(i)$
$fchild(i)$	first child of node i	$i+1$ if i is not a leaf
$lchild(i)$	last child of node i	$open(close(i)-1)$ if i is not a leaf
$nsibling(i)$	next sibling of node i	$close(i)+1$ (if the result j holds $B[j]=0$ then i has no next sibling)
$psibling(i)$	previous sibling of node i	$open(i-1)$ (if $B[i-1]=1$ then i has no previous sibling)
$subtree(i)$	number of nodes in the subtree of node i	$(close(i)-i+1)/2$
$levelancestor(i,d)$	ancestor j of i such that $depth(j)=depth(i)-d$	$bwdsearch(i,-d-1)+1$
$levelnext(i)$	next node of i with the same depth	$fwdsearch(close(i),1)$
$levelprev(i)$	previous node of i with the same depth	$open(bwdsearch(i,0)+1)$
$levelleftmost(d)$	leftmost node with depth d	$fwdsearch(0,d)$
$levelrightmost(d)$	rightmost node with depth d	$open(bwdsearch(2n+1,d))$
$lca(i,j)$	the lowest common ancestor of two nodes i, j	$parent(rmq(i,j)+1)$ unless $isancestor(i,j)$ (so $lca(i,j)=i$) or $isancestor(j,i)$ (so $lca(i,j)=j$)
$deepestnode(i)$	the (first) deepeest node in the subtree of i	$rMq(i,close(i))$
$height(i)$	the height of i (distance to its deepest node)	$excess(deepestnode(i))-excess(i)$
$degree(i)$	number of children of node i	$mincount(i+1,close(i)-1)$
$child(i,q)$	qth child of node i	$minselect(i+1,close(i)-1,q-1)+1$
$childrank(i)$	number of siblings to the left of node i	$mincount(parent(i)+1,i)+1$ unless $B[i-1]=1$ (so $childrank(i)=1$)
$leafrank(i)$	number of leaves to the left and up to node i	$rank_{10}(i)$
$leafselect(k)$	kth leaf of the tree	$select_{10}(k)$
$numleaves(i)$	number of leaves in the subtree of node i	$leafrank(close(i))-leafrank(i-1)$
$leftmostleaf(i)$	leftmost leaf of node i	$leafselect(leafrank(i-1)+1)$
$rightmostleaf(i)$	rightmost leaf of node i	$leafselect(leafrank(close(i)))$

Table 1: Operations on ordinal trees, where i and j are node identifiers.